Fast Stabbing of Boxes in High Dimensions

Franck Nielsen

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Fast Stabbing of Boxes in High Dimensions

Franck Nielsen

N° 2854

Avril 1996

THÈME 2

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

Fast Stabbing of Boxes in High Dimensions

Franck Nielsen

Thème 2 — Génie logiciel
et calcul symbolique
Projet Prisme

Rapport de recherche n ° 2854 — Avril 1996 — 30 pages

Abstract: We present in this technical report a simple yet efficient algorithm for stabbing a set $\mathcal{S}$ of n axis-parallel boxes in d-dimensional space with c points in output-sensitive time $O(dn + n \log c)$ and linear space. Let c^* be the minimum number of points required to stab $\mathcal{S}$, then we prove that

$$c \leq \min\{\frac{{c^*}^{\overline{d}}}{d!} + \frac{{c^*}^{\overline{d-1}}}{(d-1)!} - 1, c^* \frac{(\log n + 1)^{\overline{d-1}}}{(d-1)!}\},$$

where $x^{\overline{m}}$ is the rising factorial power: $x^{\overline{m}} = \prod_{i=0}^{m-1}(x+i) = \binom{x+m-1}{m}$. Since finding a minimal set of c^* points is NP-complete as soon as $d > 1$, we obtain a fast precision-sensitive heuristic for stabbing $\mathcal{S}$ in output-sensitive time and linear space. In the case of congruent or 'constrained' isothetic boxes, our algorithm reports respectively $c \leq 2^{d-1}c^*$ and $c = O(c^*)$ stabbing points. Moreover, we show that the bounds we get on c are tight and corroborate our results with some experiments. We also describe an optimal output-sensitive algorithm for finding a minimal-size optimal stabbing point-set of intervals. Finally, we conclude with insights for further research.

Key-words: Computational geometry, Output-sensitive algorithms.

(Résumé : tsvp)

INRIA, BP93, 06902 Sophia-Antipolis cedex (France) E-mail: Franck.Nielsen@sophia.inria.fr — http://www.inria.fr:/prisme/personnel/nielsen/frank.html, and University of Nice Sophia-Antipolis, Valrose (France).

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Un Algorithme Efficace pour Percer un Ensemble de Boîtes en Grandes Dimensions

Résumé : Nous présentons dans ce rapport de recherche un algorithme simple et efficace pour percer un ensemble $\mathcal{S}$ de n boîtes isothétiques de dimension d. Notre algorithme calcule un ensemble de c points perçant $\mathcal{S}$ en temps sensible à la sortie $O(dn + n \log c)$ et espace linéaire. Si c^* est le nombre minimal de points requis pour percer $\mathcal{S}$, alors nous montrons que $c \leq \min\{\frac{c^{*\overline{d}}}{d!} + \frac{c^{*\overline{d-1}}}{(d-1)!} - 1c^* \frac{(\log n+1)^{\overline{d-1}}}{(d-1)!}\}$, ou $x^{\overline{m}}$ est la puissance factorielle croissante: $x^{\overline{m}} = \prod_{i=0}^{m-1}(x+i) = \binom{x+m-1}{m}$. Puisque trouver un ensemble minimal de c^* points est un problème NP-complet dès que $d > 1$, nous obtenons une heuristique adaptative efficace pour percer $\mathcal{S}$ en temps sensible à la sortie et espace linéaire. Dans le cas de boîtes isothétiques congruentes ou de boîtes isothétiques contraintes, notre algorithme renvoie au plus $c \leq 2^{d-1}c^*$ et $c = O(c^*)$ points. De plus, nous prouvons que les bornes obtenues sur c sont précises et corroborons nos résultats théoriques par des performances pratiques. Nous décrivons également un algorithme adaptatif optimal qui calcule un ensemble de taille minimale de points perçant une famille d'intervalles. Finalement, nous concluons sur les aspects ultérieurs de ce travail.

Mots-clé : Géométrie algorithmique, Algorithmes adaptatifs.

1 Setting the problem

Let $\mathcal{S}$ be a set of n d-dimensional geometric objects of constant descriptive-size. We say that $\mathcal{S}$ is stabbed by k points if there exist k points so that each object of $\mathcal{S}$ contains at least one of these points. Thus, $\mathcal{S}$ can be immobilized under translation with k points. Given a set $\mathcal{S}$ as above, finding the minimum k so that $\mathcal{S}$ can be stabbed by k points has been shown to be NP-complete [FPT81] as soon as $d \geq 2$[1]. Therefore this problem is untractable for small values of n (say $n \simeq 20$ and $d = 2$ – See section 3.4). This problem is also referenced in the literature as the *set covering problem* (or dually as the *hitting set problem*) where it is transformed into an optimization problem by means of matrix formulations. Let $\mathcal{V} = \{S_i | i \in I\}$ be a collection of $v = |\mathcal{V}| = |I|$ subsets of $2^{\mathcal{S}}$ for a set $\mathcal{S}$ of n elements. We want to find a minimal covering collection, i.e. a sub-set $I' \subseteq I$ of indices so that $\mathcal{S} = \bigcup_{i \in I'} S_i$ with $|I'|$ as small as possible. In other words, we want to minimize $e^T \times x = |I'|$ subject to $Ax \geq e$ for x a $\{0,1\}^n$-vector, $e = (1, ..., 1)$ and A a $(v \times n)$-binary matrix, each column of which is the incidence vector of one of the sets I_i, $1 \leq i \leq v$.

Some heuristics that give approximation of the minimum stabbing number c^* have been given. V. Chvátal [Chv79] gave a polynomial time (cubic) greedy algorithm to find a cover set of size c such that $c \leq c^*(1 + \log k)$ where k is the maximum column sum ($k \leq n$). D.S. Hochbaum [Hoc82] proposed another cubic algorithm with a cover set of size at most $c^* f$, where f is the maximum row sum. Interestingly, Bellare et al. [BGLR93] showed that no polynomial time algorithm can approximate the optimal solution within a factor of $(\frac{1}{8} - \epsilon) \log |\mathcal{S}|$, unless $NP \subseteq DTIME[n^{\log \log n}]$, where $\epsilon > 0$. This result has been recently succesfully extended to the best possible $\log n$ bound by U. Feige [Fei96].

One major drawback from the computational geometry point of view is that these methods do not consider geometrical objects nor their shapes. (Although it has been shown that the intersection graph[2] of d dimensional convex objects can be arbitrary as soon as $d \geq 3$ [Weg67]). This means that we have to supply matrix A. One way to proceed is to consider from the whole arrangement of the objects all the sets defined by vertices. More precisely, to each vertex we associate the set of objects containing it (thus, the size of the matrix is $O(n^d) \times n$ and these algorithms require $O(n^{d+2})$ time and $O(n^{d+1})$ space).

[1] More precisely, Fowler et al. [FPT81] showed that covering a set of points with fixed-size squares (the so-called BOX-COVER problem) is NP-complete as soon as $d > 1$.

[2] The intersection graph of a set of objects is defined as follows: we associate to each object a node and there exists an edge between two nodes iff the corresponding objects intersect.

D.S. Hochbaum and W. Maass [HM84] considered the case of geometrical objects and give polynomial approximation scheme (note that no fully approximation scheme exists unless P=NP — we refer the reader to the comprehensive text book [GJ79] for a complete explanation). Their method is innovative since it is general and takes into account the nature of the objects. It should be noted that their method applies only to geometric objects. Unfortunately, the running time of these algorithms are at least cubic and thus cannot handle a huge amount of data. Moreover, their algorithm considers sets of identical convex objects T, or dually covering sets of points with convex translates T^*. (T^* is the centrally symmetric convex object of T).

Many applications coming from VLSI design, image processing and point location have to deal with large inputs [TF80]. Recently, H. Brönninman and M.T. Goodrich [BG94] investigate these problems using the Vapnik-Chervonenkis dimension (VC-dimension). They obtain precision-sensitive set covers if the VC-dimension[3] is bounded as it is generally the case when considering geometric objects. Their algorithm uses nontrivial concepts (and subroutines) such as set systems, ϵ-net, net finder, ... (see also [Mat91]) and still relies on the fact that matrix A is computed beforehand.

In this paper, we are even more restrictive by considering the case of axis-parallel boxes in high dimensions (that are often considered in VLSI design, image processing and point location in d-dimensional euclidean space); for example, we are given a set of points in $\mathbb{E}^d$ and some hypercube H_d. We want to associate to each point a hypercube that contains it so that we minimize the number of hypercubes. In other words, we want to cover the point set with a minimum number of patches, i.e. translates of H_d. Throughout the paper, the boxes are considered to be closed, i.e. points on the boundary of box B stab B. Our main algorithm, described in section 3, will not require to compute the arrangement of the isothetic boxes[4]. Note that, we do not consider d as a constant in the sequel.

We give in this paper a simple algorithm and study its approximation factor[5].

This fast algorithm may be useful in many applications. More precisely, we give a truly output-sensitive $O(dn + n \log c)$-time algorithm that computes a set of c points stabbing the set of n d-dimensional boxes. Interestingly, we show that $c \leq \min\{\frac{c^{*\overline{d}}}{d!} + \frac{c^{*\overline{d-1}}}{(d-1)!} - 1, c^* \frac{(\log n+1)^{\overline{d-1}}}{(d-1)!}\}$ where c^* is the optimal value and $x^{\overline{m}}$ is the rising factorial power: $x^{\overline{m}} = \prod_{i=0}^{m-1}(x+i) = \binom{x+m-1}{m}$ and $x^{\overline{0}} = 1$. Moreover, we

[3] The VC-dimension of d-dimensional isothetic boxes is 2.

[4] Computing the arrangement of a set of n isothetic boxes cost $O(n^d)$ time and space [PS85].

[5] α is an *approximation factor* of an algorithm A if $c \leq \alpha c^*$ where c^* is the optimal value and c is the value delivered by algorithm A.

exhibit a generic example where this bound is matched. We can refine the complexity analysis to show that $c \leq 2^{d-1}c^*$ and $c = O(c^*)$ when dealing respectively with congruent isothetic boxes and 'constrained' boxes.

The paper is organized as follows:

In section 2, we consider the case of a family of n intervals and give an optimal $\Theta(n(\log c^* + 1))$-time algorithm that gives an optimal stabbing set of c^* points. We use this basic case in order to devise another algorithm and analyze its behavior.

In section 3, we enhance the algorithm in higher dimensional space and study both its running time and its approximation factor. We show that the given bounds are tight. We refine the analysis for sets of congruent isothetic and constrained isothetic boxes. We corroborate our theoritical results with experiments.

Finally, in section 4, we conclude and give several guidelines for future research.

2 An optimal algorithm for stabbing intervals

In this section, we consider the case of intervals, i.e. 1-dimensional boxes. Let $\mathcal{S}$ be a set of n intervals.

2.1 Principle

Finding the minimum value c^* so that $\mathcal{S}$ can be stabbed with c^* points is easy and already known in [DG82, HM84]. Consider the interval I that has the rightmost left endpoint p. I must be stabbed by a point and clearly, the best place to stab it is on the left endpoint p. We then remove all the intervals stabbed by p and loop until all the intervals are stabbed. We thus obtain a minimal-size set of c^* points that stab $\mathcal{S}$. A straightforward algorithm based on these facts has running time $O(nc^*)$ with linear space. We show below how an adequate preprocessing can yield an optimal output-sensitive algorithm in time $\Theta(n(\log c^* + 1))$.

Remark 1. In dimension 2 (and therefore in higher dimensions), the rectangle R with the rightmost left edge is not necessary the one that has the topmost bottom edge so that we cannot exhibit a rectangle R where we can easily *a priori* compute the best place to pierce it.

Remark 2. Let $b^*(\mathcal{S})$ be the maximum size of any subset of pairwise disjoint boxes of $\mathcal{S}$. Clearly, $c^*(\mathcal{S}) \geq b^*(\mathcal{S})$. In the 1-dimensional case, the above algorithm shows that $c^*(\mathcal{S}) = b^*(\mathcal{S})$ (which is not the case in higher dimensions). This property

is no longer true in higher dimensions. (For example, in dimension 2, we may have $c^*(\mathcal{S}) \geq \frac{3}{2} b^*(\mathcal{S})$.)

2.2 Getting an output-sensitive algorithm

The methodology consists in grouping the intervals into groups and to preprocess each group in order to answer efficiently queries [Cha95, NY95]. Typically, our queries are of two kinds: "what are the intervals stabbed by a point p?" and "which interval has the rightmost left endpoint?". Moreover, we must be able to remove some of these intervals at some steps. We use the interval tree of McCreight [McC80, PS85] as the data structure for answering these queries. Assume we know an estimate $\tilde{c^*}$ of c^*. Then, we group the n intervals into $\lceil \frac{n}{\tilde{c^*}} \rceil$ groups of size at most $\tilde{c^*}$ and preprocess each group into a static interval tree[6] for a total cost of $O(\lceil \frac{n}{\tilde{c^*}} \rceil \tilde{c^*} \log \tilde{c^*}) = O(n \log \tilde{c^*})$. At some step i, we find the rightmost left endpoint p_i of the remaining set of intervals and remove the n_i intervals stabbed by p_i from their corresponding groups. Thus the total cost of this step is $O(n_i \log \tilde{c^*} + \frac{n}{\tilde{c^*}} \log \tilde{c^*})$ (see [PS85] pp 352–355). Therefore, the total cost of these c^* steps is $O(n \log \tilde{c^*} + \frac{n}{\tilde{c^*}} c^* \log \tilde{c^*})$ time since $\sum_{i=1}^{c^*} n_i = n$. If we only want to know if $c^* > p$ we can derive a $O(n \log p)$-time algorithm by choosing $\tilde{c^*} = p$ and stopping the iterative process as soon as we have computed $i = \min\{c^*, p\}$ stabbing points. Note that our algorithm works in time $O(n(\log c^* + 1))$ if $c^* \leq \tilde{c^*} < {c^*}^2$. Since we do not know c^* beforehand we iteratively estimate it by squaring our current estimate. We start with any arbitrary value for $\tilde{c^*}$, say $\tilde{c^*} = 2$. Thus, we obtain an $O(\sum_{i=0}^{\lceil \log \log c^* \rceil} n 2^i) = O(n \log c^*)$ time algorithm.

Since verifying if among n numbers k are distinct requires $\Omega(n \log k)$ time on the real RAM [KS86], it follows that this lower bound also holds for the stabbing problem by reduction in linear time. Therefore, we obtain the following theorem:

Theorem 1 *Given a set $\mathcal{S}$ of n intervals, there exists an optimal output-sensitive algorithm that reports an optimal stabbing point set of size c^* in optimal $\Theta(n \log c^*)$ time with linear space.*

Remark. As a direct consequence, we obtain a $\Theta(n \log c^*)$-time algorithm for computing the union of a n-interval set $\mathcal{S}$, where c^* is the minimal number of points required to stab $\mathcal{S}$. Note that it is not possible to get an $O(n \log C(\mathcal{S}))$-time algorithm for computing the union of intervals, where $C(\mathcal{S})$ is the number of connected components of $\mathcal{S}$. (We may have $c^*(\mathcal{S}) = \lceil \frac{n}{2} \rceil$ but $C(\mathcal{S}) = 1$.)

[6] In this context, static means that we know beforehand the $2\tilde{c^*}$ endpoints of each group. We only remove intervals and not add new ones to that data-structure.

Table 1: The table below shows the value of i as a function of c^*. We estimate c^* by $\tilde{c^*} = 2^{2^i}$. For example, for $2^{16} = 65536 \leq c^* < 2^{32}$, we set $\tilde{c^*}$ to 2^{2^4}.

$c^* =$	4	16	256	65536	2^{32} (10 digit-number)
i ($\tilde{c^*} = 2^{2^i}$) $= 0$	1	2	3	4	

2.3 A fast algorithm and its analysis

One major drawback when implementing the previous algorithm is the preprocessing step (or *guessing step* when we find a good estimate $\tilde{c^*}$ of c^*) that takes into account only a few critical values, namely the values in $\{2^{2^i} | i \in \mathbb{N}\}$. (That is the width $2^{2^i}(2^{2^i} - 1)$ of the i-th range, where the group size does not change, is rapidly growing and therefore the preprocessing step becomes inefficient in practice). We only make at most 4 guessing steps in practice — See Table 1.

Moreover, we have $c^* \leq \tilde{c^*} < {c^*}^2$ so that in the worst-case, we choose $\tilde{c^*} = {c^*}^2 - 1$ in order to adapt our preprocessing in time $O(n) \log {c^*}^2 = 2O(n) \log c^*$. This yields a factor of 2 in the hidden constant. By drawing the graph of the complexity function, we see that it is always under the graph of a function $an \log c^*$ for some real a and that the function is linear by intervals. The size of these intervals however grows quadratically and explains the poor behavior in practice of that algorithm.

We propose below a very simple "divide & conquer" algorithm :

Basic case. If $n = 1$ then choose the left endpoint of $I \in \mathcal{I} = \{I\}$ for piercing $\mathcal{I}$.

Recurse. Find the median m of the $2n$ endpoints and partition $\mathcal{I}$ depending on whether the intervals contain m ($\mathcal{I}_m$), are located to the left of m ($\mathcal{I}_1$) or to the right of m ($\mathcal{I}_2$). Choose point m to stab the intervals of $\mathcal{I}_m$ and recurse on $\mathcal{I}_1$ and $\mathcal{I}_2$.

Let $c^*(\mathcal{I})$ denote the minimum number of points required for stabbing $\mathcal{I}$ and denote by $c(\mathcal{I})$ the number of points returned by the algorithm. Observe that $\mathcal{I}$ can be stabbed with a single point iff there are n left endpoints followed by n right endpoints. Let x_n and x_{n+1} be respectively the n-th and $(n+1)$-th smallest endpoints of the $2n$ endpoints. Then, we compute our median $m = \frac{x_n + x_{n+1}}{2}$ in linear time [BFP+72].

Let us prove that $c(\mathcal{I}) \leq 2c^*(\mathcal{I}) - 1$.

Proof. We have:

$$c(\mathcal{I}) = \begin{cases} c(\mathcal{I}_m) = c^*(\mathcal{I}) = 1 & \text{if both } \mathcal{I}_1 \text{ and } \mathcal{I}_2 \text{ are empty,} \\ 1 + c(\mathcal{I}_1) + c(\mathcal{I}_2) & \text{otherwise.} \end{cases}$$

We can modelize the running of the algorithm with a binary tree. Each node has a subset of intervals $\mathcal{I}_m$ and a point stabbing $\mathcal{I}_m$. Its left (right) child is $\mathcal{I}_1$ (resp. $\mathcal{I}_2$). Now, if $\mathcal{I}_1 \neq \emptyset$ and $\mathcal{I}_2 \neq \emptyset$ then we have at least two disjoint intervals. A leaf of this tree is a set of intervals that are stabbed by a single point. Thus, the number of stabbing points reported by our algorithm is the number of nodes of that tree. The number of leaves of the tree is bounded by the maximum number b^* of pairwise disjoint intervals. Clearly, $c^*(\mathcal{I}) \geq b^*$. The tree is a strictly binary tree, i.e. each internal node has exactly two children. Thus, there are $b^* - 1$ internal nodes and $c(\mathcal{I}) \leq 2b^* - 1 \leq 2c^*(\mathcal{I}) - 1$.

The trick is to ensure that whenever $\mathcal{I}_1 \neq \emptyset$ (resp. $\mathcal{I}_2 \neq \emptyset$) then $\mathcal{I}_2 \neq \emptyset$ (resp. $\mathcal{I}_1 \neq \emptyset$). This comes from the fact that we have $|\mathcal{I}_1| = |\mathcal{I}_2|$.

Note that we can take $m = x_n$ instead of $m = \frac{x_n + x_{n+1}}{2}$. We obtain the same result.

□

One may ask whether this bound is tight or not. We can prove the tightness of this result by building a family $\mathcal{I}$ of intervals such that $c(\mathcal{I}) = 2c^*(\mathcal{I}) - 1$: Let $\mathcal{I}_1 = \{[0,1], [\frac{1}{2}, \frac{5}{2}], [2,3]\}$. We have $c(\mathcal{I}_1) = 3 = 2 * c^*(\mathcal{I}_1) - 1$. Then, we can recursively set in the intervals $[\frac{1}{2}, 1]$ and $[2, \frac{5}{2}]$ another 'scaled' copy of $\mathcal{I}_1$ and so on, ... Thus, we are able to build a family $\mathcal{I}$ of $3 \times (2^i - 1)$ intervals (for any $i \geq 1$) so that $c(\mathcal{I}) = 2c^*(\mathcal{I}) - 1 = 2^{i+1} - 1$. Indeed, we have by construction $c^*(\mathcal{I}_{i+1}) = c^*(\mathcal{I}_i) + c^*(\mathcal{I}'_i)$, $c^*(\mathcal{I}_i) = c^*(\mathcal{I}'_i)$, $c(\mathcal{I}_i) = c(\mathcal{I}'_i)$ and $c(\mathcal{I}_{i+1}) = c(\mathcal{I}_i) + c(\mathcal{I}'_i) + 1 = 2c(\mathcal{I}_i)$ where $\mathcal{I}'_i$ is a translated copy of $\mathcal{I}_i$. The proof follows by induction on i.

Let $t(\mathcal{I})$ denote the running time of the algorithm ($n = |\mathcal{I}|$). Then, we have:

$$t(\mathcal{I}) = \begin{cases} An & \text{if } c(\mathcal{I}) = 1, \\ Bn + t(\mathcal{I}_1) + t(\mathcal{I}_2) & \text{otherwise.} \end{cases}$$

with A, B some constants. We can prove by induction on n that $t(\mathcal{I}) \leq Cn \log c + Cn$ where $c = c(\mathcal{I})$ and $C = \max\{A, B\}$.

Proof.

Clearly, if $n = 1$ then $t(\mathcal{I}) = A \leq C \log c + C$ (note that $c = 1$). Otherwise, either $t(\mathcal{I}) = An \leq Cn \log c + Cn$ if $c(\mathcal{I}) = c^*(\mathcal{I}) = 1$ or

$$t(\mathcal{I}) \leq Bn + Cn' \log c_1 + Cn' \log c_2 + 2Cn',$$

with $n' \leq \frac{n}{2}$ and $c_1 + c_2 + 1 = c$. Thus,

$$t(\mathcal{I}) \leq Bn + C\frac{n}{2} \max_{c_1+c_2=c} \{\log c_1 + \log c_2\} + Cn.$$

The last inequality is maximized for $c_1 = c_2 \leq \frac{c}{2}$ since the logarithmic function is concave[7]. Therefore, we have:

$$t(\mathcal{I}) \leq Bn + Cn \log c - Cn + Cn \leq Cn \log c + Cn.$$

□

Theorem 2 *Let $\mathcal{S}$ be a family of n intervals and denote by c^* the minimal number of points required to stab $\mathcal{S}$. Then, there exists a simple output-sensitive algorithm that computes a set of c points stabbing $\mathcal{S}$ in time $O(n \log c)$ and linear space, with $c \leq 2c^* - 1$.*

Remark 1. If we pick a randomly chosen endpoint x of $\mathcal{I}$ and set $m = x$ then we obtain $c(\mathcal{I}) \leq (1 + \lceil \log n \rceil)c^*(\mathcal{I})$ instead of the output-sensitive bound $c(\mathcal{I}) \leq 2c^*(\mathcal{I}) - 1$.

Remark 2. We may get a (truly) fast algorithm that reports an optimal stabbing point set by first running the above approximate algorithm and getting a set of c points so that $c^* \leq c \leq 2c^* - 1$. In a second step, we choose $\tilde{c^*} = c$ and run the optimal algorithm described in section 2.2. Thus, the graph of the complexity function of this new algorithm is not anymore linear by intervals.

Remark 3. Although the greedy algorithm [Chv79] performs generally a H_n factor from the optimum ($H_n = \sum_{i=1}^{n} \frac{1}{i}$ and $H_n \leq 1 + \log n$), for the case of intervals we can show that it will return at most $2c^* - 1$ points [Nie96]. Moreover, there is a family of intervals where the greedy algorithm attains this worst-case bound.

3 The algorithm in higher dimensions

3.1 Principle

The "divide-and-conquer" strategy holds in any dimension and for any kind of objects. We show in that section how we can get results on the approximation factor when dealing with axis-parallel boxes. Let $\mathcal{S}$ be a set of n d-dimensional boxes. Each

[7] Clearly, if f is a concave function then we have $f(a) + f(b) \leq 2f(\frac{a+b}{2})$.

box can be viewed as the intersection of $2d$ halfspaces. A facet f of a box B is a $(d-1)$-dimensional box of the boundary ∂B supported by a hyperplane H_f. If H_f is defined by an equation of type $x_i = l$ for some real l then we say that f is a facet of *type* i. In other words, a facet of type i is perpendicular to the i-th axis. A box B can be viewed as the ordered cartesian product $\prod_{i=1}^{d}[r_i^-(B), r_i^+(B)]$ where $[r_i^-(B), r_i^+(B)]$ is the range of B along the i-th dimension. We say that box B is to the *left* (*right*) of $x_i = l$ if $r_i^+(B) < l$ (resp. $r_i^-(B) > l$). Let $X^{(d)}$ be the set of values defining the facets of type d, i.e. $X^{(d)} = \{x | \exists\ B \in \mathcal{S}$ such that $r_d^-(B) = x$ or $r_d^+(B) = x\}$.

We describe below the algorithm (see also Figure 1):

Intervals (Basic case). If $\mathcal{S}$ is one-dimensional then apply the optimal algorithm of section 2.2 for piercing this set of intervals.

Partition. Let $x_n^{(d)}$ and $x_{n+1}^{(d)}$ be respectively the n-th and $(n+1)$-th greatest elements of $X^{(d)}$. Compute the 'median' $m = \frac{x_n^{(d)} + x_{n+1}^{(d)}}{2}$ of $X^{(d)}$ in linear time [BFP+72]. Partition $\mathcal{S}$ according to the hyperplane $H_m : (x_d = m)$;

- Let $\mathcal{S}_1$ be the set of boxes that do not cross H_m and are to the left of H_m.
- Let $\mathcal{S}_2$ be the set of boxes that do not cross H_m and are to the right of H_m.
- Let $\mathcal{S}_m$ be the set of boxes intersecting H_m.

Marriage. Stab the boxes of $\mathcal{S}_m$ by piercing the set of $(d-1)$-dimensional boxes: $\mathcal{S}'_m = \{B \cap H_m | B \in \mathcal{S}_m\}$.

Conquest. Stab recursively $\mathcal{S}_1$ and $\mathcal{S}_2$.

Let $t(\mathcal{S})$ and $c(\mathcal{S})$ be respectively the running time of the algorithm and the number of stabbing points delivered by this algorithm. Sometimes, when we want to specify the dimension d of $\mathcal{S}$, we put in subscript of these notations a d. Thus, $t_d(\mathcal{S})$ and $c_d(\mathcal{S})$ denote respectively the running time and the output size of our algorithm for a set of d-dimensional isothetic boxes S. Denote by $c^*(\mathcal{S})$ the minimum number of stabbing points of set $\mathcal{S}$. We study both the approximation factor and the running time of the algorithm. In the sequel, d is not assumed to be a constant.

Our algorithm relies on the following simple facts:

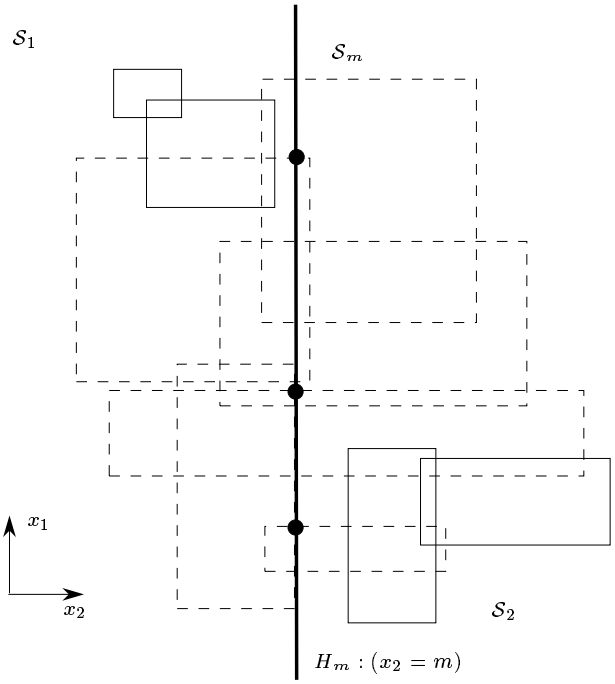

Figure 1: Partition of $\mathcal{S}$ into three subsets depending on their location with respect to the hyperplane $H_m : (x_d = m)$. We denote by $\mathcal{S}'_m$ the set of $(d-1)$-dimensional boxes $\mathcal{S}_m \cap H_m$.

Monotonicity. For any O, $c^*(\{O\} \cup \mathcal{S}) \geq c^*(\mathcal{S})$.

Additive rule. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two subsets so that $\forall I_1 \in \mathcal{I}_1,\ \forall I_2 \in \mathcal{I}_2, I_1 \cap I_2 = \emptyset$ then $c^*(\mathcal{I}_1 \cup \mathcal{I}_2) = c^*(\mathcal{I}_1) + c^*(\mathcal{I}_2)$.

Cutting rule. Let $\mathcal{S}$ be a set of boxes and H a hyperplane of type i, with $1 \leq i \leq d$. Then, $c^*(\mathcal{S}_H) = c^*(\mathcal{S}'_H)$ where $\mathcal{S}'_H = \{B \cap H | B \in \mathcal{S}\}$ and $\mathcal{S}_H = \{B | B \cap H \neq \emptyset\}$. This rule seems to be appropriate only for isothetic d-boxes.

We have:

$$c_d(\mathcal{S}) = \begin{cases} c_{d-1}(\mathcal{S}_m \cap H_m) & \text{if } \mathcal{S}_1 = \mathcal{S}_2 = \emptyset, \\ c_{d-1}(\mathcal{S}_m \cap H_m) + c_d(\mathcal{S}_1) + c_d(\mathcal{S}_2) & \text{otherwise.} \end{cases}$$

Let us prove by induction on the lexicographically ordered vector (d, n) that $c \leq \frac{c^*(\mathcal{S})^{\overline{d}}}{d!} + \frac{c^*(\mathcal{S})^{\overline{d-1}}}{(d-1)!} - 1$.

Proof.

For $d = 1$, section 2.2 describes an optimal algorithm so that $c(\mathcal{S}) = c^*(\mathcal{S}) \leq c^*(\mathcal{S}) + 1 - 1$ since $x^{\overline{0}} = 1$ by convention (finite calculus rules may be found in [GKP94]). If $|\mathcal{S}| = n = 1$ then $c(\mathcal{S}) = c^*(\mathcal{S}) = 1 \leq \frac{1^{\overline{d}}}{d!} + \frac{1^{\overline{d-1}}}{(d-1)!} - 1$.

Otherwise ($d > 1$ and $n > 1$), there are two cases depending on whether $\mathcal{S}_1, \mathcal{S}_2 = \emptyset$ or not (recall that $|\mathcal{S}_1| = |\mathcal{S}_2|$).

If $\mathcal{S}_1 = \mathcal{S}_2 = \emptyset$ then we have:

$$c(\mathcal{S}) \leq \frac{c^*(\mathcal{S})^{\overline{d-1}}}{(d-1)!} + \frac{c^*(\mathcal{S})^{\overline{d-2}}}{(d-2)!} - 1 \leq \frac{c^*(\mathcal{S})^{\overline{d}}}{d!} + \frac{c^*(\mathcal{S})^{\overline{d-1}}}{(d-1)!} - 1,$$

since $\frac{(c^*(\mathcal{S})+d-1)(c^*(\mathcal{S})+d-2)}{d(d-1)} \geq 1$ (recall that $c^*(\mathcal{S}) \geq 1$).

If $\mathcal{S}_1, \mathcal{S}_2 \neq \emptyset$ then we have:

$$c_d(\mathcal{S}) = c_d(\mathcal{S}_1) + c_d(\mathcal{S}_2) + c_{d-1}(\mathcal{S}'_m),$$

$$c(\mathcal{S}) \leq \frac{c^*(\mathcal{S}_1)^{\overline{d}} + c^*(\mathcal{S}_2)^{\overline{d}}}{d!} + \frac{c^*(\mathcal{S}_1)^{\overline{d-1}} + c^*(\mathcal{S}_2)^{\overline{d-1}}}{(d-1)!} + \left(\frac{c^*(\mathcal{S})^{\overline{d-1}}}{(d-1)!} + \frac{c^*(\mathcal{S})^{\overline{d-2}}}{(d-2)!}\right) - 3,$$

with $1 \leq c^*(\mathcal{S}_1), c^*(\mathcal{S}_2)$ and $c^*(\mathcal{S}_1) + c^*(\mathcal{S}_2) \leq c^*(\mathcal{S})$ since $c^*(\mathcal{S}'_m) = c^*(\mathcal{S}_m) \leq c^*(\mathcal{S})$. Since the rising factorial power is convex, the right hand side of the last inequality

is maximized[8] for $c^*(\mathcal{S}_1) = 1$ and $c^*(\mathcal{S}_2) = c^*(\mathcal{S}) - 1$ (or the other way around). It follows that:

$$c(\mathcal{S}) \leq \frac{(c^*(\mathcal{S})-1)^{\overline{d}}}{d!} + \frac{(c^*(\mathcal{S})-1)^{\overline{d-1}}}{(d-1)!} + (\frac{c^*(\mathcal{S})^{\overline{d-1}}}{(d-1)!} + \frac{c^*(\mathcal{S})^{\overline{d-2}}}{(d-2)!}) - 3 + \frac{1^{\overline{d}}}{d!} + \frac{1^{\overline{d-1}}}{(d-1)!},$$

$$c(\mathcal{S}) \leq \frac{c^*(\mathcal{S})^{\overline{d-1}}}{d!}(c^*(\mathcal{S}) - 1 + d) + \frac{c^*(\mathcal{S})^{\overline{d-2}}}{(d-1)!}(c^*(\mathcal{S}) - 1 + d - 1) - 1,$$

and therefore

$$c(\mathcal{S}) \leq \frac{c^*(\mathcal{S})^{\overline{d}}}{d!} + \frac{c^*(\mathcal{S})^{\overline{d-1}}}{(d-1)!} - 1.$$

□

The tightness of the analysis will allow us to derive a collection of examples in section 3.2 such that $c(\mathcal{S}) = \frac{c^*(\mathcal{S})^{\overline{d}}}{d!} + \frac{c^*(\mathcal{S})^{\overline{d-1}}}{(d-1)!} - 1$.

Remark. For small values of $c^*(\mathcal{S})$ we get an approximation factor which is polynomial in d. As an example, consider $c^*(\mathcal{S}) = 3$ then we have $c(\mathcal{S}) \leq d(d+2)$ (the approximation scheme of Hochbaum and Maass [HM84] has an exponential dependence in d).

However, when $c^*(\mathcal{S})$ is large (say $c^*(\mathcal{S}) > dn^{\frac{1}{d}}$) we have the trivial bound $c(\mathcal{S}) \leq n$. It does not reflect the dichotomy process. Therefore, we study the relationships between $c(\mathcal{S})$ and $|\mathcal{S}| = n$.

Let us prove that:

$$c(\mathcal{S}) \leq c^*(\mathcal{S}) \frac{(\log n + 1)^{\overline{d-1}}}{(d-1)!}. \qquad (\star)$$

Proof.

Recall that we denote by $c_d(\mathcal{S})$ the number $c(\mathcal{S})$ of stabbing points returned by our algorithm for a set of d-dimensional isothetic boxes $\mathcal{S}$ (this is an abuse of notation that might however help the reader when checking the proofs).

[8] Let $f(\cdot)$ be a convex function defined on range $[a, b]$ then $\max_{a \leq c \leq b}\{f(c)\} \leq \max\{f(a), f(b)\}$. Let $g_m(x) = x^{\overline{m}}$. $g_m(x)$ is convex on $[0, +\infty)$ for $m \geq 0$. Define $f(x) = g_m(x) + g_m(c-x) + g_{m-1}(x) + g_{m-1}(c-x)$ for $1 \leq x \leq c-1$. $f(x)$ is convex on $[1, c-1]$ and therefore $\max_{1 \leq x \leq c-1}\{f(x)\} \leq f(1)$.

We do the induction on the lexicographically ordered vector (d, n). If $d = 1$ then section 2.2 gave an optimal algorithm, i.e. $c(\mathcal{S}) = c_1(\mathcal{S}) = c^*(\mathcal{S})$ and $(\star)$ holds trivially since $x^{\overline{0}} = 1$ and $0! = 1$. If $n = 1$ then $c(\mathcal{S}) = c^*(\mathcal{S}) = 1 \leq 1 \times \frac{1^{\overline{d-1}}}{(d-1)!}$.

Otherwise ($d > 1$ and $n > 1$), we distinguish on whether $\mathcal{S}_1, \mathcal{S}_2 = \emptyset$ or not. In the former case, we have:

$$c(\mathcal{S}) = c_{d-1}(\mathcal{S}) \leq c^*(\mathcal{S}) \frac{(\log n + 1)^{\overline{d-2}}}{(d-2)!} \leq c^*(\mathcal{S}) \frac{(\log n + 1)^{\overline{d-1}}}{(d-1)!},$$

since $\frac{\log n + 1 + d - 2}{d-1} \geq 1$ with equality for $n = 1$.

In the latter case, we have:

$$c_d(\mathcal{S}) = c_d(\mathcal{S}_1) + c_d(\mathcal{S}_2) + c_{d-1}(\mathcal{S}'_m),$$

with $|\mathcal{S}_1| = |\mathcal{S}_2| \leq \frac{n}{2}$ and $c_{d-1}(\mathcal{S}'_m) = c_d(\mathcal{S}_m) \leq c^*(\mathcal{S}) \frac{(\log n + 1)^{\overline{d-2}}}{(d-2)!}$. Thus, we get

$$c_d(\mathcal{S}) \leq \Big(c^*(\mathcal{S}_1) + c^*(\mathcal{S}_2) \Big) \frac{(\log \frac{n}{2} + 1)^{\overline{d-1}}}{(d-1)!} + c^*(\mathcal{S}) \frac{(\log n + 1)^{\overline{d-2}}}{(d-2)!},$$

$$c_d(\mathcal{S}) \leq c^*(\mathcal{S}) \Big(\frac{(\log n)^{\overline{d-1}}}{(d-1)!} + \frac{(\log n + 1)^{\overline{d-2}}}{(d-2)!} \Big),$$

since $c^*(\mathcal{S}_1) + c^*(\mathcal{S}_2) = c^*(\mathcal{S}_1 \cup \mathcal{S}_2) \leq c^*(\mathcal{S})$.

$$c(\mathcal{S}) \leq c^*(\mathcal{S}) \frac{(\log n + 1)^{\overline{d-2}}}{(d-1)!} (\log n + d - 1),$$

$$c(\mathcal{S}) \leq c^*(\mathcal{S}) \frac{(\log n + 1)^{\overline{d-1}}}{(d-1)!}.$$

□

Let us now analyze the time spent by this algorithm for reporting the $c_d(\mathcal{S})$ stabbing points.

We have:

$$t_d(\mathcal{S}) = \begin{cases} 0 & \text{if } \mathcal{S} = \emptyset, \\ An + t_{d-1}(\mathcal{S}'_m) + t_d(\mathcal{S}_1) + t_d(\mathcal{S}_2) & \text{otherwise.} \end{cases}$$

with A some constant related to the implementation of the partition scheme [BFP+72] (e.g., a typical value for A is 4 – see [BFP+72]).

Clearly, a trivial bound we can get is $t_d(\mathcal{S}) \leq Adn(2c(\mathcal{S}) - 1)$:

We prove below by induction on the lexicographically ordered vector (d, n) that $t_d(\mathcal{S}) \leq A(n \log c(\mathcal{S}) + dn)$ $(\star)$.

Proof. If $d = 1$ then we proved in section 2.2 an $O(n \log c(\mathcal{S}) + n)$-time algorithm. Therefore, $t_1(S) \leq Bn(\log c(\mathcal{S}) + 1) \leq An(\log c(\mathcal{S}) + 1)$ for $A \geq B$. If $n = 1$ then $t_d(\mathcal{S}) \leq Ad$ and $(\star)$ holds trivially.

Otherwise ($d > 1$ and $n > 1$), consider the two cases depending on whether $\mathcal{S}_1, \mathcal{S}_2 = \emptyset$ or not: In the former case, we have $t_d(\mathcal{S}) = t_{d-1}(\mathcal{S}) + An$. Thus, we get

$$t_d(\mathcal{S}) \leq An\Big((d-1) + \log c(\mathcal{S})\Big) + An \leq An(d + \log c(\mathcal{S})).$$

In the latter case ($\mathcal{S}_1, \mathcal{S}_2 \neq \emptyset$) , we have:

$$t_d(\mathcal{S}) = An + t_d(\mathcal{S}_1) + t_d(\mathcal{S}_2) + t_{d-1}(\mathcal{S}'_m),$$

with $1 \leq |\mathcal{S}_1|, |\mathcal{S}_2| \leq \frac{n}{2}$, $|\mathcal{S}_1| + |\mathcal{S}_2| + |\mathcal{S}'_m| = n$ and $c(\mathcal{S}) = c(\mathcal{S}_1) + c(\mathcal{S}_2) + c(\mathcal{S}'_m)$. Let $n' = |\mathcal{S}_1| = |\mathcal{S}_2|$ ($|\mathcal{S}'_m| = n - 2n'$). Hence,

$$t_d(\mathcal{S}) \leq A\Big(n + n' \log c(\mathcal{S}_1) + n'd + n' \log c(\mathcal{S}_2) + n'd + (n-2n') \log c(\mathcal{S}'_m) + (n-2n')(d-1)\Big),$$

$$t_d(\mathcal{S}) \leq A\Big(nd + 2n' + n'(\log c(\mathcal{S}_1) + \log c(\mathcal{S}_2)) + (n - 2n') \log c(\mathcal{S}'_m)\Big).$$

But $\log c(\mathcal{S}_1) + \log c(\mathcal{S}_2)$ is maximized when $c(\mathcal{S}_1) = c(\mathcal{S}_2) \leq \frac{c(\mathcal{S})}{2}$.

Therefore, we get

$$t_d(\mathcal{S}) \leq A\Big(dn + 2n' \log c(\mathcal{S}) + (n - 2n') \log c(\mathcal{S}'_m)\Big),$$

with $c(\mathcal{S}'_m) \leq c(\mathcal{S})$.

Finally, we get $t_d(\mathcal{S}) \leq An(d + \log c(\mathcal{S}))$.

□

Note that when $c(\mathcal{S}) = c^*(\mathcal{S}) = 1$ then our algorithm requires $O(nd)$-time (this is easily checked). Let $N = O(nd)$ denote the input size of our set $\mathcal{S}$ of n d-dimensional boxes. Then, our algorithm runs in $O(N + \frac{N}{d} \log c)$-time and linear space.

Theorem 3 *Let $\mathcal{S}$ be a set of n d-dimensional boxes. Denote by c^* the minimum number of stabbing points of $\mathcal{S}$. Then, there exists an output-sensitive algorithm that reports a set of c stabbing points of $\mathcal{S}$ in time $O(dn + n \log c)$ with linear space whose approximation c is bounded by* $\min\{\frac{{c^*}^{\overline{d}}}{d!} + \frac{{c^*}^{\overline{d-1}}}{(d-1)!} - 1, c^* \frac{(\log n+1)^{\overline{d-1}}}{(d-1)!}\}$.

Remark 1. It would be interesting to analyze probabilistically the algorithm under the uniform distribution of the isothetic d-dimensional boxes [Cof88]. Note that if the stabbing points are split into two balanced groups with respect to the hyperplane H_{m_s} at each stage s of the algorithm, then we get $c(\mathcal{S}) \leq c^*(\mathcal{S}) \frac{(\log c^*(\mathcal{S})+1)^{\overline{d-1}}}{(d-1)!}$. We expect in practice far better results. This is mainly due to the fact that we maximize all the terms in the complexity analysis (e.g., $c(\mathcal{S}'_m), c(\mathcal{S}_1), c(\mathcal{S}_2)$) – See section 3.4. Indeed, it is unlikely that $c^*(\mathcal{S}'_m) = c^*(\mathcal{S})$.
Remark 2. Extending this approach to the case of a family of $n \geq 2$ d-dimensional k-oriented objects (for a fixed constant k), we get an $O(dn + n \log c)$-time algorithm which is however not anymore precision-sensitive (this is mainly due to the fact that the Helly number of these families is $d + 1$ and not anymore 2 as for the case of parallelotopes, i.e. axis-parallel boxes). In that case, we can prove that $c \leq (k \log n + 1)^{d-1}$.
Remark 3. As soon as we consider $d - boxes$ with $d \geq 2$, greedy algorithm may yield a H_n factor from the optimum [Nie96].

3.2 A bad example

Let us now analyze the tightness of the upper bounds. In that section we assume w.l.o.g. that $m = x_n^{(d)}$ (see the algorithm in section 3.1).

Consider first the planar case. We want to exhibit an example showing that our algorithm reports $\Omega(\min\{{c^*}^2, 1 + c^* \log n\})$ stabbing points for a family of planar boxes (we proved, for the planar case, that $c(\mathcal{S}) \leq \min\{c^*(\mathcal{S}) \frac{c^*(\mathcal{S})+3}{2} - 1, c^*(\mathcal{S})(1 + \log n)\}$). In the following, we define a box by its S.W. (leftmost bottommost) and N.E. (rightmost uppermost) corners.

Let E_{c^*} be a family of boxes requiring exactly c^* points to be stabbed. Denote by $|E_{c^*}|$ its cardinality, i.e. the number of boxes of E_{c^*}. We use a parameter b in order to build recursively E_{c^*} (initially, $b = 1$ and $\forall b, b'$, $|E_{c^*}(b)| = |E_{c^*}(b')|$, in other words $|E_{c^*}(b)|$ does not depend on b).

Define $E_{c^*}(b)$ recursively as follows (Figure 2 depicts the construction of $E_{c^*}(b)$):

- $E_1(b) = [(0, \frac{3}{4}), (b, \frac{5}{4})]$.

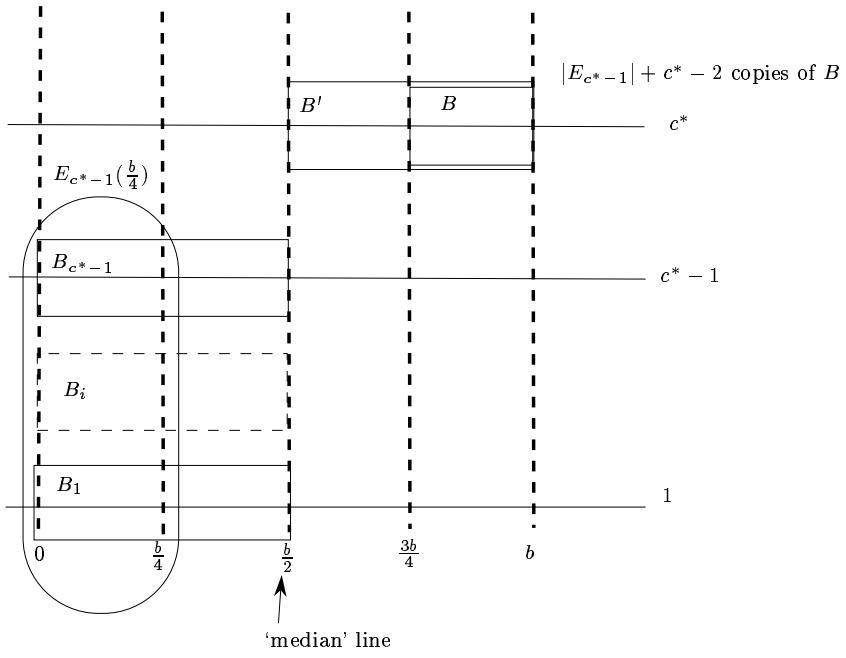

Figure 2: Building a family $E_{c^*}(b)$ of 2-boxes so that $c(E_{c^*}(b)) = \Omega(c^*(E_{c^*}(b))^2)$.

- $|E_{c^*-1}| + c^* - 2$ copies of box $B = [(\frac{3b}{4}, c^* - \frac{1}{4}), (b, c^* + \frac{1}{4})]$ union $\{B_i = [(0, i - \frac{1}{4}), (\frac{b}{2}, i + \frac{1}{4})] | 1 \leq i < c^*\}$ union box $B' = [(\frac{b}{2}, c^* - \frac{1}{4}), (b, c^* + \frac{1}{4})]$ union the boxes resulting from $E_{c^*-1}(\frac{b}{4})$ (note that b becomes $\frac{b}{4}$ at this stage).

Clearly, $E_{c^*}(b)$ can be stabbed with a minimal set of c^* points: $\left\{(i, \frac{3b}{4^{c^*-i+1}}) | 1 \leq i \leq c^*\right\}$.

But our algorithm will report exactly $(\sum_{i=1}^{c^*} i) + c^* - 1 = \frac{c^*(c^*+1)}{2} + c^* - 1 = \Omega({c^*}^2)$ points.

Note that in order to design this example, we need at least $n = 2^{c^*-1}$ boxes. (Indeed, we have $|E_{c^*}| = 2|E_{c^*-1}| + 2(c^* - 1)$ and $E_1 = 1$). So that we also check that $c \leq c^*(1 + \log n)$.

We can also build in higher dimensions, using the same principle, a family $\mathcal{S}$ of boxes so that $c(\mathcal{S}) = \Omega(\frac{c^*(\mathcal{S})^d}{d!})$. Recall that a d-dimensional box B can be viewed as an ordered product of d ranges: $B = [a_1, b_1] \times ... \times [a_d, b_d]$. Given a $(d-1)$-dimensional box B', we can extend it to a d-dimensional box $B = [a_1, b_1] \times B'$.

Define the set $E_{c^*}^{(d)}(b)$ of boxes as follows:

- $E_1^{(d)}(b) = [(0, \frac{3}{4}, ..., \frac{3}{4}), (b, \frac{5}{4}, ..., \frac{5}{4})]$ for any d.
- $|E_{c^*-1}^{(d)}| + |E_{c^*}^{(d-1)}| - 1$ copies of box $B = [(\frac{3b}{4}, ..., \frac{3b}{4}, c^* - \frac{1}{4}), (b, ..., b, c^* + \frac{1}{4})]$ union $[0, \frac{b}{2}] \times E_{c^*}^{(d-1)}(\frac{b}{2})$ union the box $B' = [(\frac{b}{2}, \ldots, \frac{b}{2}, c^* - \frac{1}{4}), (b, \ldots, b, c^* + \frac{1}{4})]$ union $E_{c^*-1}^{(d)}(\frac{b}{4})$.

Clearly, we have $|E_{c^*}^{(d)}| = 2(|E_{c^*}^{(d-1)}| + |E_{c^*-1}^{(d)}|)$.

$E_{c^*}^{(d)}(b)$ can be pierced with an optimal set of c^* points: $\left\{(i, \frac{3b}{4^{c^*-i+1}}, ..., \frac{3b}{4^{c^*-i+1}}) | 1 \leq i \leq c^*\right\}$. However, our algorithm will return $\Omega(\frac{{c^*}^d}{d!})$ stabbing points (using Euler's summation formula). More precisely, we can prove by induction on d that our algorithm will return exactly $c = \frac{{c^*}^d}{d!} + \frac{{c^*}^{d-1}}{(d-1)!} - 1$ stabbing points.

Proof. We do the proof by induction on the lexicographically ordered vector (d, n). If $d = 1$, cleary $c(\mathcal{S}) = c^*(\mathcal{S})$. If $|\mathcal{S}| = n = 1$ then $c(\mathcal{S}) = c^*(\mathcal{S}) = 1$. Otherwise, following our algorithm we have:

$$c(E_{c^*}^{(d)}) = 1 + c(E_{c^*-1}^{(d)}) + c(E_{c^*}^{(d-1)}).$$

Plugging our inductive hypothesis, we get:

$$c(E_{c^*}^{(d)}) = 1 + (\frac{c^*(E_{c^*-1}^{(d)})^{\overline{d}}}{d!} + \frac{c^*(E_{c^*-1}^{(d)})^{\overline{d-1}}}{(d-1)!} - 1) + (\frac{c^*(E_{c^*}^{(d-1)})^{\overline{d-1}}}{(d-1)!} + \frac{c^*(E_{c^*}^{(d-1)})^{\overline{c-2}}}{(d-2)!} - 1),$$

using $c^*(E_{c^*}^{(k)}) = c^*$ for any $k \geq 1$, we get

$$c(E_{c^*}^{(d)}) = \frac{c^{*\overline{d-1}}}{d!}(c+d-1) + \frac{c^{*\overline{d-2}}}{(d-1)!}(c+d-2) - 1.$$

Setting back $c^* = c^*(E_{c^*}^{(d)})$, we finally obtain the desired result:

$$c(E_{c^*}^{(d)}) = \frac{c^*(E_{c^*}^{(d)})^{\overline{d}}}{d!} + \frac{c^*(E_{c^*}^{(d)})^{\overline{d-1}}}{(d-1)!} - 1.$$

□

We can also check that $c \leq c^* \frac{(\log n+1)^{\overline{d-1}}}{(d-1)!}$ since $2^{c^*-1} \leq |E_{c^*}^{(d)}| \leq 4^{c^*}$ (more precisely $|E_{c^*}^{(d)}| = O(2^{c^*} c^{*d-1})$).

Thus, when the dimension d is fixed, we have exhibited a family of boxes such that the algorithm gives an approximation that matches the upper bound $\frac{c^{*\overline{d}}}{d!} + \frac{c^{*\overline{d-1}}}{(d-1)!} - 1$. We can also provide the same kind of scheme to build a family $\mathcal{S}$ of unit hypercubes so that $c(\mathcal{S}) = \Omega(\frac{c^*(\mathcal{S})^{\overline{d}}}{d!})$.

3.3 Congruent or Constrained boxes

The previous section exhibits an example where our algorithm reaches its worst-case performance. However, in order to build it, we did consider stretched boxes, i.e. non-constrained boxes. We define the aspect ratio of a d-dimensional box $B = [(a_1, ..., a_d), (b_1, ..., b_d)]$ to be $\max_{i,j=1..d}\{\frac{b_i - a_i}{b_j - a_j}\}$. Loosely speaking, we will call aspect ratio of $\mathcal{S}$ the maximum of the aspect ratio of the boxes in $\mathcal{S}$. Note that if the boxes have bounded aspect ratio then also have bounded-size (volume) but not the converse. Thus, hypercubes have aspect ratio equal to 1. We use nonbounded aspect ratio in order to build our bad example. In [HM84], the polynomial-time approximation scheme depends on the aspect ratio of the congruent isothetic boxes.

In this section, we refine the analysis of the approximation factor whenever the projections of the d-boxes onto the d axis have the bounded aspect ratio property.

Note that the boxes may have a nonbounded aspect ratio but their projections (sets of intervals) may have their aspect ratio bounded.

Lemma 4 *Let $\mathcal{S}$ be a set of n congruent isothetic d-dimensional boxes. Then, our algorithm guarantees that $c(\mathcal{S}) \leq 2^{d-1}c^*(\mathcal{S})$.*

Proof. For sake of simplicity, let us first consider the case of (unit) hypercubes. $\mathcal{S}$ is a collection of n congruent hypercubes. We prove below by induction on the dimension that $c(\mathcal{S}) \leq 2^{d-1}c^*(\mathcal{S})$. Section 2.2 shows that the algorithm ensures $c_1(\mathcal{S}) = c^*(\mathcal{S})$ for (unit) intervals.

Otherwise, let $\mathcal{S}$ be a set of n d-boxes. Consider the ordered sequence (left to right) of cutting hyperplanes of type d: $(H_m(1) : x_d = a_1), ..., (H_m(k) : x_d = a_k)$ with the associated partition of the hypercubes $\mathcal{S}'_m(1), ..., \mathcal{S}'_m(k)$. Clearly, we have $a_{i+1} - a_i > \frac{1}{2}, 1 \leq i \leq k-1$ for the case of unit hypercubes. Therefore $\mathcal{S}'_m(i) \cap \mathcal{S}'_m(j) = \emptyset$ if $|i-j| \geq 2$. We have:

$$c_d(\mathcal{S}) = \sum_{i=1}^{k} c_{d-1}(\mathcal{S}'_m(i)),$$

$$c_d(\mathcal{S}) \leq 2^{d-2} \sum_{i=1}^{k} c^*(\mathcal{S}'_m(i)).$$

We can decompose the last sum taking into account the parity of i:

$$c_d(\mathcal{S}) \leq 2^{d-2} \Big(\sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} c^*(\mathcal{S}'_m(2i)) + \sum_{i=0}^{\lceil \frac{k}{2} \rceil - 1} c^*(\mathcal{S}'_m(2i+1)) \Big).$$

But $\sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} c^*(\mathcal{S}'_m(2i)) = c^*(\cup_{i=1..\lfloor \frac{k}{2} \rfloor} \mathcal{S}'_m(2i)) \leq c^*(\mathcal{S})$ and $\sum_{i=0}^{\lceil \frac{k}{2} \rceil - 1} c^*(\mathcal{S}'_m(2i+1)) = c^*(\cup_{i=0..\lceil \frac{k}{2} \rceil - 1} \mathcal{S}'_m(2i+1)) \leq c^*(\mathcal{S})$ since both $S'_m(2i) \cap S'_m(2j) = \emptyset$ and $S'_m(2i+1) \cap S'_m(2j+1) = \emptyset$ as soon as $i \neq j$.

Therefore, we get:

$$c_d(\mathcal{S}) \leq 2^{d-2} \times 2c^*(\mathcal{S}),$$

$$c_d(\mathcal{S}) \leq 2^{d-1}c^*(\mathcal{S}).$$

We only use the fact that all the boxes have the same i-width (width along the i-th dimension). Therefore, the result applies for congruent boxes. □

In [HM84], Hochbaum and Maass also consider this problem (in its dual form however) and gave an $O(l^d n^{2l^d+1})$-time algorithm (a polynomial time approximation scheme) which ensures that $c(\mathcal{S}) \leq (1+\frac{1}{l})^d c^*(\mathcal{S})$ for a given integer $l \geq 1$. Thus, for $l = 1$ it yields an $O(n^3)$-time algorithm with performance ratio 2^d.

Since our algorithm proceeds dimension by dimension, we only need to have the bounded aspect ratio for the projected boxes (along the d axis). Let $\mathcal{S}$ be a collection of n constrained boxes:

$$\mathcal{S} = \{[(b_{1,1}, ..., b_{1,d}), (u_{1,1}, ..., u_{1,d})], ..., [(b_{n,1}, ..., b_{n,d}), (u_{n,1}, ..., u_{n,d})]\},$$

with $\max_{i,j=1..n}\{\frac{u_{i,j}-b_{i,1}}{u_{j,1}-b_{j,1}}\} \leq B_1$, ..., $\max_{i,j=1..n}\{\frac{u_{i,d}-b_{i,d}}{u_{j,d}-b_{j,d}}\} \leq B_d$, for some constants $B_i \geq 1$, $1 \leq 1 \leq d$.

Using the same technic as above, we get the following lemma:

Lemma 5 *Let $\mathcal{S}$ be a collection of n d-dimensional constrained boxes with $B_1, ..., B_d$ defined as above. Then, our algorithm will return $c(\mathcal{S})$ stabbing points so that $c(\mathcal{S}) \leq (\prod_{i=2}^{d} \lceil 2B_i \rceil) c^*(\mathcal{S})$.*

We may assume w.l.o.g. that $B_1 = \max_{i=1..d}\{B_i\}$. Otherwise, we make a simple rotation of the orthogonal frame in linear time. This also means that we may have a direction where the projected boxes are not constrained since we are able to solve exactly the problem in one dimension (see Section 2.2).

We can also mix up our algorithm with the PTAS (polynomial-time approximation scheme) of [HM84] in order to obtain tradeoffs both for the running time and the performance ratio.

One might wonder whether an additive constant is possible instead of a multiplicative constant in all our bounds concerning $c(\mathcal{S})$. It is very unlikely! Indeed, unless P=NP is it not possible to achieve the absolute performance $c(\mathcal{S}) \leq c^*(S) + k$ for some constant integer k.

Proof. We do the proof by contradiction (see also [GJ79] for this kind of technic). Consider that we have a deterministic polynomial time algorithm ALG that guarantees $c(\mathcal{S}) \leq c^*(\mathcal{S}) + k$. Then we consider $k+1$ non-overlapping copies of our boxes. Let $\mathcal{S}_{k+1}$ be this data set. Algorithm ALG will return $c(\mathcal{S}_{k+1})$ stabbing points so that $c(\mathcal{S}_{k+1}) \leq c^*(\mathcal{S}_{k+1}) + k$. One of the copies of $\mathcal{S}$ in $\mathcal{S}_{k+1}$ requires at most $\lfloor \frac{c(\mathcal{S}_{k+1})}{k+1} \rfloor = c(\mathcal{S})$ stabbing points and $c^*(\mathcal{S}_{k+1}) = (k+1)c^*(\mathcal{S})$. Thus, we will be able to have a deterministic polynomial-time algorithm ALG-B such that

$c^*(\mathcal{S}) \leq c(\mathcal{S}) \leq c^*(\mathcal{S}) + \frac{k}{k+1}$. Since finding the value of $c^*(\mathcal{S})$ is NP-complete, we get the contradiction unless $P = NP$. □

Let $\mathcal{G}$ be the intersection graph of a set of n d-dimensional isothetic boxes $\mathcal{S}$, i.e. to each box corresponds a node and there is an edge between two nodes iff their corresponding boxes intersect. Isothetic boxes have nice combinatorial properties. For example, a set of boxes have a nonempty intersection iff they intersect pairwise (this is an Helly-type theorem [DG82, HD60]). Therefore, finding a minimum-size set of stabbing points can be done by first computing $\mathcal{G}$ in quadratic time and then, finding a minimum clique partition of $\mathcal{G}$, i.e. a set of cliques (complete subsets of $\mathcal{G}$) whose union covers the vertices of $\mathcal{G}$. There is a one-to-one correspondence between these two problems. As a direct corollary, it implies that the mimimum clique partition is NP-complete (as proved in [GJ79]), even for intersection graphs of isothetic boxes.

Note that if every 3-subset of $\mathcal{S}$ (i.e., a subset $S \in \binom{\mathcal{S}}{3}$) has an empty intersection ($\cap S = \emptyset$), then the stabbing problem can be solved in *polynomial time* by reduction to the maximum matching problem.

The stabbing problem is related somehow to $c(p,q)$-numbers and $N(p,q;d)$ numbers [HD60, GW93]. $N(p,q;d)$ numbers are defined for $2 \leq q \leq p$ for the class $\mathcal{P}$ of isothetic d-boxes as follows: it is the smallest integer so that for every set $\mathcal{S} \in \mathcal{P}$ of parallelotopes, we have: if every p-subset $S \in \binom{\mathcal{S}}{p}$ contains at least one q-subset $Q \in \binom{S}{q}$ that has a nonempty intersection ($\cap Q \neq \emptyset$), then $\mathcal{S}$ can be stabbed by $N(p,q;d)$ points. Debrunner et al. [HD60, GW93] proved that $N(p,q;d) \leq \binom{p-q+d}{d}$ if $2 \leq q \leq p$.

Therefore, we have:

$$c^* \leq N(b^*+1, 2; d) = \binom{b^* - 1 + d}{d},$$

since every (b^*+1)-subset of $\mathcal{S}$ contains at least a pair of intersecting boxes.

There is much literature concerning $c(p,q)$ numbers. Perhaps, one of the most challenging conjecture that remains unsettled is the following:

Conjecture (Wegner 67). If $\mathcal{K}$ is a family of parallel rectangles in the plane, no p of which are pairwise disjoint, then $\mathcal{K}$ can be stabbed by $2p-3$ points.

This conjecture has been confirmed for squares and $p \leq 4$ but remain opened for 'nonconstrained' boxes, i.e. non-bounded size boxes.

3.4 Experimental results

We did the implementation in C++ using the LEDA[9] and CGAL[10] librairies. The code length is about 1000 lines. It should be noted that the algorithm and therefore its implementation are robust. Indeed, we only compare our standard input values without creating intermediate values (even when our algorithm reports intersection points). Since the problem of computing the minimum value c^* so that our set is c^*-pierceable is NP-complete, we could not compare in the experiments the precise relationships between c and c^*. Note that if instead of implementing the optimal stabbing algorithm of intervals of section 2.2, we implement the heuristic of section 2.3 then we get $c \leq O(\binom{c^*+d-1}{d})$. We did choose that solution in our experiments (see discussion of section 2.3).

However, loosely speaking, if we admit that in average $c^*(\mathcal{S}'_m) = c^*(\mathcal{S})^{\frac{d-1}{d}}$ and $c^*(\mathcal{S}_1) = c^*(\mathcal{S}_2)$ (which might be the case, for example, when considering the uniform distribution of congruent boxes) then we expect $c(\mathcal{S})$ to be an $O(1)$-approximate of $c^*(\mathcal{S})$ for fixed dimension d. Indeed, let K_d with $K_1 = 1$ be the multiplicative constant. We have:

$$c(\mathcal{S}) \leq K_d(c^*(\mathcal{S})) \leq c^* + \sum_{i=0}^{\lceil \log c^*(\mathcal{S}) \rceil} 2^i K_{d-1}((\frac{c^*(\mathcal{S})}{2^i})^{\frac{d-1}{d}}).$$

Therefore, we find $K_d \leq 1 + \frac{1}{2^{\frac{1}{d}}-1} K_{d-1}$ with $K_1 = 1$. Thus, in the planar case we get under these hypothesis that $c(\mathcal{S}) \leq 3.42c^*(\mathcal{S})$. (This result is corroborated in the experiments (using a good lower bound for $c^*(\mathcal{S})$, e.g. $b(\mathcal{S})$).

We report the value of $c(\mathcal{S})$ in Table 2 for a set $\mathcal{S}$ of uniformly distributed isothetic planar boxes. We have also implemented an exhaustive search procedure for finding a minimal set of piercing points. This algorithm could not handle input size greater than 25 (although some tricks have been plugged[11] in). Table 3 shows the number of configurations explored by the exhaustive algorithm.

As an application, we considered the following problem (already mentionned in the introduction): given a set of n cities $\mathcal{C} = \{C_1 = (x_1, y_1), ..., C_n = (x_n, y_n)\}$, we wish to cover them by a minimum number of a given 'unit' square $S = [-r, r] \times [-r, r]$

[9] Library for Efficient DataStructure and Algorithms. Max-Planck Institut für Informatik, Im Stadtwald – 66123 Saarbrücken – Germany.

[10] see "The CGAL Kernel User Manual" – INRIA Sophia-Antipolis (France).

[11] Finding good cuts for an exhaustive algorithm is interesting in itself since it allows to handle large input sizes for 'special' down-to-earth tailored instances (see for example the well-known Traveling Salesman Problem [RSL74]).

$\lvert\mathcal{S}\rvert$ =	16	32	64	128	256	512	1024	2048	4096	8192	16364	32728	65536	130000	260000	520000
$c(\mathcal{S})$ =	9	14	23	30	56	78	123	188	272	417	648	928	1413	2093	3122	4486

Table 2: Considering uniformly distributed rectangles in the unit square.

$\lvert\mathcal{S}\rvert$ =	14	15	16	17	18
conf($\mathcal{S}$) =	15120	29400	99120	241272	672964

Table 3: Number of different configurations conf($\mathcal{S}$) for some input set $\mathcal{S}$, i.e. number of distinct piercing point sets of size ranging over $[c^*, n]$ for small values of n. Note that for d-dimensional set $\mathcal{S}$ of boxes, the complexity of the arrangement of $\mathcal{S}$ is $O(|\mathcal{S}|^d)$. Thus, we have $\text{conf}(\mathcal{S}) \leq \sum_{i=c^*(\mathcal{S})}^{|\mathcal{S}|} \binom{O(|\mathcal{S}|^d)}{i} \leq 2^{O(|\mathcal{S}|^d)}$.

of side length $2r$. We associate to each city $C_i = (x_i, y_i)$ the square $S_i = [x_i - r, x_i + r] \times [y_i - r, y_i + r]$ for $1 \leq i \leq n$. Let $\mathcal{S} = \{S_1, ..., S_n\}$. Now, we use the fact that $\mathcal{C}$ can be covered with k translated copies of S if and only if $\mathcal{S}$ can be stabbed with k points. We ran our program on 120 cities of the United States of America (the input file is freely distributed and may be found in the *Stanford GraphBase* [Knu93]). The results are depicted in figure 3.

Figure 4 shows some experiments for sets $\mathcal{S}$ that are 20-pierceable, i.e. $c^*(\mathcal{S}) = 20$. The left chart shows the number of stabbing points reported by our algorithm in case of congruent/nonconstrained boxes. The right chart depicts the running time of our algorithm. (We took the average over 20 trials).

It may be nice to study the average stabbing number $a_n(d)$ of a fixed-size randomly chosen set of n d-boxes as a function of d.

4 Concluding remarks

We have investigated in this paper the *stabbing problem* for a set of d-dimensional isothetic boxes which consists in finding a set of points so that each box contains at least one of these points.

Finding a minimum-size set of stabbing points has been shown to be NP-complete as soon as $d > 1$, even when considering congruent isothetic boxes. Therefore

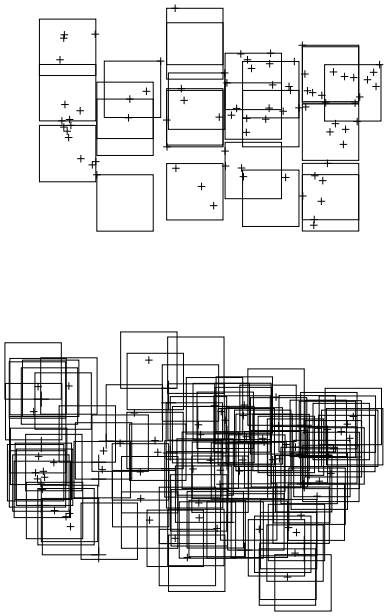

Figure 3: Finding a covering of 120 cities of the U.S.A. Our algorithm reports a set of 25 unit squares covering the 120 cities. Any solution requires at least 11 squares. (Considering the corresponding piercing problem, one may find 11 disjoint squares.)

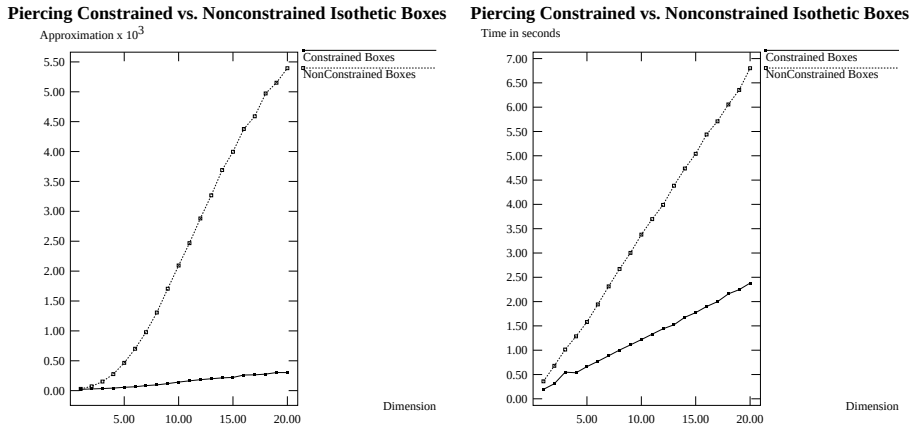

Figure 4: Impact of the dimension over a set $\mathcal{S}$ ($c^*(\mathcal{S}) = 20$ and $|\mathcal{S}| = 20000$) of constrained/nonconstrained d-boxes for $1 \leq d \leq 20$. The right chart exhibits the running time of our implementation (the slope of the lines for the constrained/nonconstrained cases are $O(d + \log c^*)$ and $O(d \log c^*)$ respectively.

that problem is untractable in practice. We gave in this technical report a heuristic that computes c points stabbing a set of n d-dimensional axis-parallel boxes in output-sensitive time $O(dn + n \log c)$ using linear space. Moreover, we proved that $c \leq \min\{\frac{c^{*\overline{d}}}{d!} + \frac{c^{*\overline{d-1}}}{(d-1)!} - 1, c^* \frac{(\log n+1)^{\overline{d-1}}}{(d-1)!}\}$, where $x^{\overline{m}}$ is the rising factorial power: $x^{\overline{m}} = \prod_{i=0}^{m-1}(x+i)$. We showed the tightness of the bounds by building a generic family $\mathcal{S}$ of d-dimensional boxes so that $c(\mathcal{S}) = \Omega(\frac{c^*(\mathcal{S})^d}{d!})$. We proved in the case of congruent boxes and 'constrained' boxes that $c \leq 2^{d-1}$ and $c = O(c^*)$ respectively. Our algorithm can be easily parallelized onto PRAM computers in order to gain efficiency (see [AL93]).

We plan to investigate in a nearby future the tradeoff between the running time of any stabbing algorithm for sets of boxes and its relative performance.

We also focus on the case of c-oriented objects and general convex/non convex objects.

This technical report raises some open problems:

- the exact relationships betwen c^* and b^* (from $N(p, q; d)$ numbers, we get $c^* \leq \binom{b^*-1+d}{d}$).

- the hardness of approximation of constrained boxes compared with general boxes inside the polynomial hierarchy of problems.
- Can we find for 'nonconstrained' isothetic boxes an algorithm which guarantees some ratio between b and b^*?
- Can we obtain better fast approximation algorithms by applying to our set of c stabbing points other (time-costly) algorithms?
- In [IA83], H. Imai and Ta. Asano gave an algorithm to compute a maximal cell in an arrangement of n isothetic boxes in $O(n^{d-1} \log n)$. Can we compute a maximal cell in a better running time? This would improve the running time of the greedy algorithm.

Another aspect of this problem that is currently being investigated is to give efficient algorithms to detect whether a set of objects is k-pierceable or not for small values of k [KN95, KN96].

Acknowledgements. I would like to thank Jean-Daniel Boissonnat, Hervé Brönnimann and Mariette Yvinec for helpful discussions.

References

[AL93] S. G. Akl and K. A. Lyons. *Parallel Computational Geometry.* Prentice-Hall, 1993.

[BFP+72] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, and Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7, 1972.

[BG94] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite vc-dimension. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 293–302, 1994.

[BGLR93] M. Bellare, S Goldwasser, C. Lund, and A. Russel. Efficient probabistically checkable proofs and applications to approximation. In *Proc. 25th Annu. Symp. Theory Comput.*, pages 294 – 304, 1993.

[Cha95] M. Y. Chan. Output-Sensitive Results on Convex Hulls, Extreme Points, and Related Problems. *Proc. of the A.C.M. Symp. on Computational Geometry*, 1995.

[Chv79] V. Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4:233–235, 1979.

[Cof88] E.G. Coffman. *Probabilistic Analysis of Packing and Partitionning Algorithms.* S. Lueker, 1988.

[DG82] L. Danzer and B. Grünbaum. Intersection Properties of Boxes in R^d. *Combinatorica*, 2(3):237 – 246, 1982.

[Fei96] U. Feige. A Threshold of $\log n$ for Approximating Set Cover. In *Proc. of the 28-th ACM Symposium on Theory of Computing*, 1996.

[FPT81] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inform. Process. Lett.*, 12(3):133–137, 1981.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York, NY, 1979.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics.* Addison-Wesley Publishing Company, New York, 1994.

[GW93] Peter M. Gruber and J. M. Wills, editors. *Handbook of Convex Geometry*, volume A. North-Holland, Amsterdam, Netherlands, 1993.

[HD60] H. Hadwiger and H. Debrunner. *Kombinatorische Geometrie in der Ebene.* Monographies de l'Enseignement Mathématique, N2, Geneva, 1960.

[HM84] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 31:130–136, 1984.

[Hoc82] D. S. Hochbaum. Approximation Algorithms of the Set Covering and Vertex Cover Problems. *SIAM J. COMPUT.*, 11(3):555 – 556, August 1982.

[IA83] H. Imai and Ta. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *J. Algorithms*, 4:310–323, 1983.

[KN95] M. J. Katz and F. Nielsen. On piercing sets of objects. Research Report à paraître, INRIA, BP93, 06902 Sophia-Antipolis, France, 1995.

[KN96] M. Katz and F. Nielsen. On Piercing Convex Sets. In *Proc. of the* 12*-th international conference on Computational Geometry*, 1996.

[Knu93] D. E. Knuth. *The Stanford Graphbase*, volume 1 of *A Platform for Combinatorial Computing.* Addison-Wesley, Reading, MA, 1993.

[KS86] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 15:287–299, 1986.

[Mat91] J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, pages 505–511, 1991. Also to appear in *J. Comput. Syst. Sci.*

[McC80] E. M. McCreight. Efficient algorithms for enumerating intersecting intervals and rectangles. Report CSL-80-9, Xerox Palo Alto Res. Center, Palo Alto, CA, 1980.

[Nie96] F. Nielsen. Note on Geometric Greedy Algorithm. In *manuscript, INRIA Sophia-Antipolis, France.*, 1996.

[NY95] F. Nielsen and M. Yvinec. Output-Sensitive Convex Hull Algorithms of Planar Convex Objects. Research Report, INRIA, BP93, 06902 Sophia-Antipolis, France, 1995.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[RSL74] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. Approximate algorithms for the traveling salesperson problem. In *Proc. 15th Annu. IEEE Sympos. Switching Automata Theory*, pages 33–42, 1974.

[TF80] S.L. Tanimoto and R.J. Fowler. Covering image subsets with patches. In *Proc. of the 5-th international conference on pattern recognition*, pages 835–839, 1980.

[Weg67] G. Wegner. *Eigenschaften der Nerven homologish-einfacher Familien im* $\mathbb{R}^n$. PhD thesis, Göttingen, 1967.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399